|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование кафедры, по которой выполняется работа |
| Дисциплина |  | Информационные технологии и программирование | | |
|  |  | наименование дисциплины | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УЧЕБНО-ПРАКТИЧЕСКАЯ РАБОТА | | | | |
|  | №1 |  |
|  | номер задания (при наличии) |  |
| РАЗРАБОТКА ИЕРАРХИИ КЛАССОВ С ИСПОЛЬЗОВАНИЕМ ИНТЕРФЕЙСОВ, | | | |
| АБСТРАКТНЫХ КЛАССОВ И ДРУГИХ МЕХАНИЗМОВ НАСЛЕДОВАНИЯ | | | |

при наличии указать тему учебно-практической работы и (или) номер варианта

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ОБУЧАЮЩИЙСЯ** | | | | | |
| группы | | | |  | О724Б |
|  |  | Кокорев М.Н. | | | |
| подпись |  | фамилия и инициалы | | | |
|  | | | | |
| дата сдачи | | | | |
| **ПРОВЕРИЛ** | | | | | |
|  | | | | | | |
| ученая степень, ученое звание, должность | | | | | | |
|  |  | Землянская Е.Р. | | | | |
| подпись |  | фамилия и инициалы | | | | |
| Оценка / балльная оценка | | |  | | |
|  | | | | |
| дата проверки | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Санкт-Петербург | | | | | | |
|  |  |  | 20 |  | г. |  |

Оглавление

[1 Постановка задачи 3](#_Toc146573179)  
[2 Реализация 6](#_Toc146573180)  
[3 Результаты работы программы: 17](#_Toc146573181)

# 1 Постановка задачи

Задание 1.

Разработать класс, согласно индивидуальному варианту, содержащий: ‒ элементы разного уровня доступа (public и private);

‒ не менее четырех свойств;

‒ не менее трех методов;

‒ перегрузку метода ToString();

‒ статический метод;

‒ константное или поле только для чтения;

‒ не менее трех конструкторов;

‒ перегрузку операции присваивания и одной любой арифметической.

Рекомендуемые поля и методы указаны в варианте. Также необходимо написать программу с меню, позволяющую протестировать разработанный класс. Обязательные пункты меню:

‒ задание параметров конструируемого объекта;

‒ вывод свойств объекта;

‒ выполнение статического метода;

‒ выполнение методов объекта.

Задание 2. Разработать иерархию классов, включающую созданный в задании 1. В иерархию должно входить минимум пять классов. Вновь добавленный корневой класс иерархии (базовый) должен быть абстрактным и содержать общие для остальных поля и методы, а хотя бы один класс из потомков – sealed. У каждого из классов должно быть хотя бы одно собственное свойство и метод. Также должны быть продемонстрированы виртуальные и переопределённые методы. Переопределенные должны вызывать методы базового класса если это оправдано. В каждом классе должен быть метод, выводящий имя данного класса. В объектах дочерних классов он должен вызывать аналогичный метод родительского класса.

Необходимо написать программу с меню, позволяющую протестировать разработанную иерархию. Обязательные пункты меню:

‒ задание свойств каждого объекта; (хотя бы по одному объекту на не абстрактный класс);

‒ вывод свойств объекта;

‒ выполнение методов объекта;

‒ вывод имени класса объекта.

Задание 3. Разработать общий интерфейс для доступа к объектам классов разработанной иерархии. Интерфейс должен предоставлять доступ к общим для всех классов иерархии методам. Необходимо добавить новый класс, не входящий в иерархию, однако также реализующий разработанный интерфейс.

Создать список объектов классов, реализующих разработанный интерфейс. Создать функцию, принимающую в качестве параметра объект любого класса, реализующего интерфейс. Написать программу с меню, позволяющую протестировать работу списка. Обязательные пункты меню:

‒ добавление нового объекта выбранного пользователем класса в список;

‒ вывод свойств объекта из списка;

‒ выполнение методов объекта из списка;

‒ вывод всех объектов в списке с указанием их классов;

‒ выполнение созданной функции с указанным объектом из списка.

Вариант 9. Железная дорога.

Класс для первой части – СВ.

Варианты свойств: количество мест, цена места, количество свободных мест, номер вагона.

Варианты методов: продать билеты, разнести чай, дать объявление о маршруте следования поезда, напомнить, что кондиционеров

нет и окна открывать нельзя (статический).

Возможные классы иерархии: вагоны (базовый), вагон-ресторан,

купе, товарный.

Возможный интерфейс: ITrain, дополнительный класс – дрезина.

# 2 Реализация

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.Design;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace \_1

{

abstract class Train : ITrain

{

string name;

public double number { get; set; }

public double cash { get; set; }

public double number\_free { get; set; }

virtual public void set\_m()

{

double w = 0; double b = 0; double C = 0;

do

{

Console.WriteLine("Количество мест: ");

} while (!Double.TryParse(Console.ReadLine(), out w) || w <= 0); number = w;

do

{

Console.WriteLine("Стоимость места: ");

} while (!Double.TryParse(Console.ReadLine(), out C) || C <= 0); cash = C;

do

{

Console.WriteLine("Свободные места: ");

} while (!Double.TryParse(Console.ReadLine(), out b) || b <= 0 || b > number); number\_free = b;

}

virtual public void get\_m()

{

Console.WriteLine(ToString());

Console.WriteLine("Кол-во мест - " + number);

Console.WriteLine("Стоимость места - " + cash);

Console.WriteLine("Кол-во свободных мест - " + number\_free);

}

virtual public void NAME()

{

Console.WriteLine("Поезд ");

}

virtual public void sell()

{

double a = 0;

Console.WriteLine("Кол-во свободных мест - " + number\_free);

do

{

Console.WriteLine("Сколько билетов продать?: ");

} while (!Double.TryParse(Console.ReadLine(), out a) || a < 0 || a > number\_free); number\_free -= a;

}

virtual public void tea()

{

double d = number - number\_free;

Console.WriteLine("Вы разнесли " + d + " чая");

}

virtual public void unique() { }

}

interface ITrain

{

void NAME();

void set\_m();

void get\_m();

void sell();

void tea();

void unique();

}

class Drezina : ITrain

{

const string name = "Дрезина";

public double number { get; set; }

public double cash { get; set; }

public double number\_free { get; set; }

override public string ToString()

{

return (name);

}

public void set\_m()

{

{

double w = 0; double b = 0; double C = 0;

do

{

Console.WriteLine("Количество мест: ");

} while (!Double.TryParse(Console.ReadLine(), out w) || w <= 0); number = w;

do

{

Console.WriteLine("Стоимость места: ");

} while (!Double.TryParse(Console.ReadLine(), out C) || C <= 0); cash = C;

do

{

Console.WriteLine("Свободные места: ");

} while (!Double.TryParse(Console.ReadLine(), out b) || b <= 0 || b > number); number\_free = b;

}

}

public void get\_m()

{

Console.WriteLine(ToString());

Console.WriteLine("Кол-во мест - " + number);

Console.WriteLine("Стоимость места - " + cash);

Console.WriteLine("Кол-во свободных мест - " + number\_free);

}

public void NAME()

{

Console.WriteLine("Поезд ");

}

public void sell()

{

double a = 0;

Console.WriteLine("Кол-во свободных мест - " + number\_free);

do

{

Console.WriteLine("Сколько билетов продать?: ");

} while (!Double.TryParse(Console.ReadLine(), out a) || a < 0 || a > number\_free); number\_free -= a;

}

public void tea()

{

double d = number - number\_free;

Console.WriteLine("Вы разнесли" + d + " чая");

}

public void unique()

{

Console.WriteLine("Вы в дрезине,(\*\_\*)");

}

}

class SW : Train

{

const string name = "СВ";

public string train\_num { get; set; }

public override void sell()

{

base.sell(); number = base.number;

}

public override void tea()

{

base.tea(); number = base.number; number\_free = base.number\_free;

}

public override void NAME()

{

base.NAME();

Console.WriteLine(name);

}

public static bool operator <(SW m1, SW m2)

{

return (m1.number < m2.number);

}

public static bool operator >(SW m1, SW m2)

{

return (m1.number > m2.number);

}

public static SW operator +(SW m1, SW m2)

{

return new SW(m1.number + m2.number, m1.cash, m1.number\_free + m2.number\_free, m1.train\_num + m2.train\_num);

}

override public string ToString()

{

return (name);

}

override public void set\_m()

{

base.set\_m(); number = base.number; cash = base.cash; number\_free = base.number\_free;

Console.WriteLine("Какой идентификатор вагона");

train\_num = Console.ReadLine();

}

override public void get\_m()

{

base.get\_m();

Console.WriteLine("Номер вагона - " + train\_num);

}

override public void unique()

{

Console.WriteLine("Вы дали объявление о маршруте поезда");

}

public static void know()

{

Console.WriteLine("напомнить, что кондиционеров нет и окна открывать нельзя");

}

public SW()

{

cash = 150;

number = 100;

number\_free = 50;

train\_num = "RUS1";

}

public SW(double number, double cash, double number\_free, string train\_num)

{

this.number = number;

this.cash = cash;

this.number\_free = number\_free;

this.train\_num = train\_num;

}

public SW(double cash, double number\_free)

{

this.cash = cash;

this.number\_free = number\_free;

train\_num = "RUS2";

number = 0;

}

}

class Rest : Train

{

const string name = "Вагон ресторан";

string colour { get; set; }

override public string ToString()

{

return (name);

}

public override void sell()

{

base.sell(); number = base.number;

}

public override void tea()

{

base.tea(); number = base.number; number\_free = base.number\_free;

}

public override void unique()

{

Console.WriteLine("Вы поели на все ваши деньги:(");

}

public override void NAME()

{

base.NAME();

Console.WriteLine(name);

}

override public void set\_m()

{

base.set\_m(); number = base.number; cash = base.cash; number\_free = base.number\_free;

Console.WriteLine("Как еда?");

colour = Console.ReadLine();

}

override public void get\_m()

{

base.get\_m();

Console.WriteLine("Вкус - " + colour);

}

}

class Kupe : Train

{

const string name = "Купе";

string neighbors { get; set; }

override public string ToString()

{

return (name);

}

public override void sell()

{

base.sell(); number = base.number;

}

public override void tea()

{

base.tea(); number = base.number; number\_free = base.number\_free;

}

public override void unique()

{

Console.WriteLine("Вы сидите (.\_.)");

}

public override void NAME()

{

base.NAME();

Console.WriteLine(name);

}

override public void set\_m()

{

base.set\_m(); number = base.number; cash = base.cash; number\_free = base.number\_free;

Console.WriteLine("Как вам соседи?");

neighbors = Console.ReadLine();

}

override public void get\_m()

{

base.get\_m();

Console.WriteLine("Соседи - " + neighbors);

}

}

sealed class Tovar : Train

{

const string name = "Товарный";

string tov { get; set; }

public override void sell()

{

base.sell(); number = base.number;

}

public override void tea()

{

base.tea(); number = base.number; number\_free = base.number\_free;

}

public override void unique()

{

Console.WriteLine("Вас не пустили внутрь, но можете попытаться еще¯l\_(ツ)\_/¯");

}

public override void NAME()

{

base.NAME();

Console.WriteLine(name);

}

override public string ToString()

{

return (name);

}

override public void set\_m()

{

base.set\_m(); number = base.number; cash = base.cash; number\_free = base.number\_free;

Console.WriteLine("Какой товар везут?");

tov = Console.ReadLine();

}

override public void get\_m()

{

base.get\_m();

Console.WriteLine("Размер - " + tov);

}

}

internal class Program

{

static void Func(ITrain item)

{

item.sell();

Console.WriteLine("Вы продали билеты");

}

static void Main(string[] args)

{

int main\_menu;

do

{

Console.Clear();

Console.WriteLine("1. 1 часть");

Console.WriteLine("2. 2 часть");

Console.WriteLine("0. Выход");

if (!Int32.TryParse(Console.ReadLine(), out main\_menu))

{

Console.WriteLine("Некорректный ввод.");

main\_menu = -1;

}

switch (main\_menu)

{

case 1: FirstTask(); break;

case 2: SecondTask(); break;

}

} while (main\_menu != 0);

void FirstTask()

{

int menu, menu2; SW m = new SW(), m2 = new SW();

do

{

Console.Clear();

Console.WriteLine("1. Добавление вашего SW");

Console.WriteLine("2. Вывод вашего SW");

Console.WriteLine("3. Выбрать выполняемый метод для SW");

Console.WriteLine("4. Сравнить SW");

Console.WriteLine("5. Совместить SW");

Console.WriteLine("0. Выход");

if (!Int32.TryParse(Console.ReadLine(), out menu))

{

Console.WriteLine("Некорректный ввод.");

menu = -1;

}

switch (menu)

{

case 1:

Console.Clear();

m.set\_m();

Console.ReadKey();

break;

case 2:

Console.Clear();

m.get\_m();

Console.ReadKey();

break;

case 3:

do

{

Console.Clear();

Console.WriteLine("1. Продать билеты");

Console.WriteLine("2. Разнести чай");

Console.WriteLine("3. Подать объявление о маршруте поезда");

Console.WriteLine("4. Подать очень важное объявление");

Console.WriteLine("0. Выход");

if (!Int32.TryParse(Console.ReadLine(), out menu2))

{

Console.WriteLine("Некорректный ввод.");

menu2 = -1;

}

switch (menu2)

{

case 1:

Console.Clear();

m.sell();

Console.ReadKey();

break;

case 2:

Console.Clear();

m.tea();

Console.ReadKey();

break;

case 3:

Console.Clear();

m.unique();

Console.ReadKey();

break;

case 4:

Console.Clear();

SW.know();

Console.ReadKey();

break;

}

} while (menu2 != 0);

break;

case 4:

Console.Clear();

Console.WriteLine("Вот параметры моего вагона СВ:");

m2.get\_m();

if (m > m2) Console.WriteLine("У вашего Св больше мест!");

else Console.WriteLine("К сожалению, ваш вагон меньше моего");

Console.ReadKey();

break;

case 5:

Console.Clear();

Console.WriteLine("Вот параметры моего СВ");

m2.get\_m();

m = m + m2;

Console.WriteLine("Ваш СВ стал блольше, за счет объединения");

Console.WriteLine("Вот новые параметры ВАШЕГО СВ:");

m.get\_m();

Console.ReadKey();

break;

}

} while (menu != 0);

}

void SecondTask()

{

int menu, menu2, i;

List<ITrain> gg = new List<ITrain>();

do

{

Console.Clear();

Console.WriteLine("1. Добавление нового вагона");

Console.WriteLine("2. Просмотр свойств вагона");

Console.WriteLine("3. Выполнение действий с вагонами");

Console.WriteLine("4. Вывод всех вагонов");

Console.WriteLine("5. Выполнение функции");

Console.WriteLine("0. Выход");

if (!Int32.TryParse(Console.ReadLine(), out menu))

{

Console.WriteLine("Некорректный ввод.");

menu = -1;

}

switch (menu)

{

case 1:

Console.Clear();

do

{

Console.Clear();

Console.WriteLine("Выберите вагон, который хотите добавить");

Console.WriteLine("1. СВ");

Console.WriteLine("2. Ресторан");

Console.WriteLine("3. Дрезина");

Console.WriteLine("4. Купе");

Console.WriteLine("5. Товарный");

Console.WriteLine("0. Выход");

if (!Int32.TryParse(Console.ReadLine(), out menu2))

{

Console.WriteLine("Некорректный ввод.");

menu2 = -1;

}

switch (menu2)

{

case 1:

Console.Clear();

SW m = new SW(); m.set\_m(); gg.Add(m);

Console.ReadKey();

break;

case 2:

Console.Clear();

Rest p = new Rest(); p.set\_m(); gg.Add(p);

Console.ReadKey();

break;

case 3:

Console.Clear();

Drezina l = new Drezina(); l.set\_m(); gg.Add(l);

Console.ReadKey();

break;

case 4:

Console.Clear();

Kupe w = new Kupe(); w.set\_m(); gg.Add(w);

Console.ReadKey();

break;

case 5:

Console.Clear();

Tovar g = new Tovar(); g.set\_m(); gg.Add(g);

Console.ReadKey();

break;

}

} while (menu2 != 0);

break;

case 2:

do

{

Console.Clear();

Console.WriteLine($"Введите номер желаемого вагона (всего объектов:{gg.Count})");

Console.WriteLine("Введите ноль для возврата)");

} while (!Int32.TryParse(Console.ReadLine(), out i) || i < 0 || i > gg.Count); i--;

if (i >= 0)

{

gg[i].get\_m();

Console.ReadKey();

}

break;

case 3:

Console.Clear();

do

{

Console.Clear();

Console.WriteLine($"Введите номер желаемого объекта (всего объектов:{gg.Count})");

Console.WriteLine("Введите ноль для возврата");

} while (!Int32.TryParse(Console.ReadLine(), out i) || i < 0 || i > gg.Count); i--;

if (i >= 0)

{

gg[i].sell();

gg[i].tea();

gg[i].unique();

Console.ReadKey();

}

break;

case 4:

Console.Clear();

foreach (var item in gg)

{

item.get\_m();

}

Console.ReadKey();

break;

case 5:

Console.Clear();

do

{

Console.Clear();

Console.WriteLine($"Введите номер желаемого объекта (всего объектов:{gg.Count})");

Console.WriteLine("Введите ноль для возврата");

} while (!Int32.TryParse(Console.ReadLine(), out i) || i < 0 || i > gg.Count); i--;

if (i >= 0)

{

Func(gg[i]);

Console.ReadKey();

}

break;

}

} while (menu != 0);

}

}

}

}

## 3 Результаты работы программы:

На рисунке 1 показан выбор задания.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рисунок 1 – Меню выбора задания

На рисунке 2 показано меню 1 задания.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 2 – Меню 1 задания

На рисунке 3 показано меню 2 и 3 заданий.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 3 – Меню 2 и 3 заданий

На рисунке 4 показано задание параметров объекта.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 4 – Задание параметров

На рисунке 5 показаны свойства созданного объекта.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок 5 – Просмотр свойств объекта

На рисунке 6 показано меню выбора метода.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 6 – Меню выбора метода

На рисунке 7 показано выполнение всех методов одного объекта.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

На рисунке 8 показано сравнение двух объектов класса СВ.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 8 – Сравнение двух объектов

На рисунке 9 показано сумма двух объектов класса CВ.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9 – Сумма двух объектов

На рисунке 10 показано меню выбора создания объекта.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок 10 – Меню создания объекта

На рисунке 11 показан вывод всех объектов списка.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 11 – Вывод всех объектов списка

На рисунке 12 показано выполнение функции для объекта.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 12 – Выполнение функции